ELEMENTARY MATLAB PROGRAMMING

(Version R2013a used here so some differences may be encountered)

INTRODUCTION

% It is assumed the user of these lessons have also had experience in MTH 229 or its equivalent and I suggest you have the latest copy of the CSI mathematics department's manual (copies available at the math department website). The early part of these notes are a review of material covered in the latter course, and even some of the later material should look familiar. Some of the same topics you have seen in MTH 229 is presented from a different perspective. It is expected that the student will refer regularly to the MTH 229 manual which does elaborate on some of the functions presented and in places you will find specific references to that material. Since MATLAB has been introduced, you should be familiar with the MATLAB environment on the CSI computer systems. Though there is a lot of repetition in these notes from your previous experience it is important to master these to be able to solve Engineering and Science problems. One major difference is the strong emphasis here on building m-files to solve the latter problems (Yes m-files are introduced in the MTH 229 notes) and introduction to writing your own function m-files) with lots of examples of such.
Some of the numerical methods topics will be presented in ENS 336 and not covered in CSC 270. In ENS 336 C++ and Excel spreadsheet techniques will be explored

% **These notes mimic commands and statements of the MATLAB Command Window. Be sure to try all commands you're are not familiar with to get a working knowledge of the MATLAB environment as you read/study the material that follows**: Review what you forgot is important.
Do All **TASK** projects as you study this workbook and hand in as assigned.
% The MATLAB environment also interprets starting a line with a % symbol as a programmer comment not to be executed by the computer. Not every comment line needed in these notes uses % it's just an indicator to you comments are being made, usually one paragraph at a time in this workbook.
% Recall the **'»'** symbol (known as the MATLAB "prompt" ie. Waiting for a command! And is used below for illustrating the commands given to MATLAB and lines following without a prompt are MATLAB responses to the command you should expect. Excess spaces you will see when executing commands from MATLAB are not shown in outputs blow to save space.

PROGRAMMING (THE BASICS)

% Programming can be thought of as the ability to write commands that the computer will carry out to help us solve problems. Most scientific and engineering problems are mathematical in nature. The MATLAB environment permits us to do powerful mathematical techniques in a relatively simple fashion. It is the function of this workbook to illustrate the basic principles of using MATLAB to solve a wide variety of numerical problems as well as introduce MATLAB as a "first" computer language for a science or
Engineering student. It is assumed the student has had calculus and maybe some physics and thus understands some of the principles of solving mathematical problems.

% A programmer needs to know how to call upon the MATLAB programming and problem solving ability. This is done by constructing various symbolic commands understood by the environment. (This is true for all programming languages). A series of commands that carry out a task is called a program.

The following presentation of programming is done via illustration.
Commands are given and MATLAB responses are shown. It is expected if a concept is causing you difficulty that you will go to the MATLAB environment and mimic the behavior of the illustrations, as well as, experiment with the concept to better enlighten yourself about its meaning.

The basic concepts needed to understand and construct programs follows.
% IT IS SUGGESTED THAT THE STUDENT PRACTICE ALL ILLUSTRATED MATLAB COMMANDS ESPECIALLY IF THEY ARE NEW TO THE PROGRAMMING WORLD!

VARIABLES

% Computer variables are storage places in the memory of a computer.
Values or the results of a calculation ARE PLACED INTO a variable. MATLAB variables are actually Matrices and Arrays (Vectors) but for starting purposes we will use them as simple single valued variables. (as most other languages do!).

% Examples of Variables and value placement

```
» x = 1      % we type
x = 1      % MATLAB echo
```

% (MATLAB echoes a response of what it did. The echo is actually over several lines but shorten them here, also spaces are added for readability)
% we asked MATLAB above to put the value 1(a constant) into a memory place called 'x'(THE VARIABLE).
% THE '=' SIGN MEANS TO ASSIGN THE VALUE TO THE VARIABLE.
% Another simple example

```
» velocity = 45.67
velocity = 45.6700
```

%Note: **values in memory are kept to many decimal. Places and the default echo is 4 places. Try LOADING VARIOUS NUMBERS TO SEE THE LATTER.**
% In the examples above the = sign can be thought of as an assignment of a value to a place in memory. The statement velocity = 45.67 means the programmer is placing the value of 45.67 into a variable called "Velocity" The computer stores this value which may be used again in a latter instruction. The programmer knows the memory section storing the value as a place called "velocity" but the computer will know the memory area as a numbered address. We might think of a single valued variable as a mailbox that we as programmers put a single value into.

VARIABLE NAMES

% MATLAB is normally case sensitive. This means that following variables are all different. Velocity, VELOCITY, velocity, VELocity. **It is recommended that you use small letters for learning programming.**
Variable names can be up to 19 characters long with a letter starting the name and letters or numbers or underscore '_' following.
Some legal names are avg, vel0, t1, end_of_year, inertia_57, etc.
Scalar variables have a single value. Later we will look at Array variables which are multivalued and can represent vectors and matrices.

SCALAR OPERATORS

% Calculations usin**g addition, subtraction, multiplication, division and exponentiation** can be done with the values in scalar variables by setting up what are called mathematical expressions with special symbols to indicate the particular mathematical operations as in c++. The symbols used are called OPERATORS. The operators presented are those appropriate for single value variables. These operators are distinguished from **ARRAY OPERATORS** for reasons that will become apparent when we study arrays**. For the time being all variables at any time have a single value and are thus called scalars**.

%

ADDITION OPERATOR +

% we set up the following sequence of commands; Note the command lines » and responses

```
» x = 1
x =    1

» y = 4
y =    4
» z = x + y
z =    5
```

% This last command z = x + y can be interpreted as follows for visualization. The computer previously stored values for x and y in its memory. It goes back to the memory and gets the values in x and y , adds them and places (assigns) the sum into the memory place z. In other languages we would need an additional command to see what was in the memory. MATLABs echo immediately shows us the result of the calculation.

```
%adding constants
» k = 45.67 + 34.54
k =   80.2100
```

%**Counting** by a series of commands

%NOTE: UP/DOWN ARROW KEY REPEATS PAST COMMANDS

```
» count = 0
count =    0

» count = count + 1
count =    1

» count = count + 1
count =    2

» count = count + 1
count =    3
» count = count + 1
count =    4
```

%We note that the value kept in the variable 'count' will climb by one, each time we execute the command count = count + 1.
We say we are **counting** by 1 in this case. We also say we **initialized** (started) the variable count with a zero.

% A **statement** of the type y = x + 1 is calculated by the computer in a fixed manner. First the mathematical expression to the right of the '=' sign is calculated. A 1 is added to What ever is in the variable x.
Second, the result of the mathematical expression is assigned into the variable, y, on the left of the = sign.
In the counting example the variable on the left happens to be the same as the one in the simple mathematical expression. ie.count.
One should note that an expression like x = x + 1 illustrates that the '=' sign in a computer statement is not an algebraic equality.   TRY TO SOLVE FOR x?
We emphasis the variable on the left will obtain the result of the evaluation of the expression on the right. A process we call an **assignment**.

% **SUBTRACTION OPERATOR -**

```
%Subtracting constants;
» v = 67.9-34.5
v =   33.4000

% Counting down;
» count =10
count =    10

» count=count-1
count =     9
» count=count-1
count =     8
» count=count-1
count =     7
» count=count-1
count =     6
```

% **MULTIPLICATION OPERATOR   ***

```
%  %multiplying constants
» y = 4 * 40
y =   160
```

% **we note if you do not provide a variable to place** the result of a calculation MATLAB automatically puts the answer in a **variable called 'ans'**
as in this next example

```
» 4 * 40
ans =   160
%multiplying variables
» x = 5
x =     5

» y = 9
y =     9
» w = x * y
w =    45

% multiplying with constants
» q = x * y * 2
q =    90
%OR
» q = 2 * y * x
q =     90
```

% We note that the order of the variables in the above case does not matter. as one would suspect by the algebra- However we have to be careful in evaluating a more complex expression.

% SIMPLE COMBINATION OF THE ABOVE OPERATORS
% **WE CAN STOP THE MATLAB ECHO BY USING A ';' AT THE END OF A LINE AS FOLLOWS**

```
» x = 4;
» y = 7;
» r = 2;
» z = x * y + r
z =    30
» z = x * y + r - 4
```

```
z =    26
```

% **EVALUATING A MATHEMATICAL EXPRESSION**

%The computer will evaluate an expression on the right of the equal sign according to strict rules of order. This is done to make it clear what the expression means algebraically. In general the expression is evaluated left to right with the following rules.

The **order of priority (evaluation order)**

1. Parentheses
2. Exponentiation (raising to a power)
3. Multiplication and Division
4. Addition and Subtraction

% **DIVISION OPERATOR /**

```
% Dividing constants
» x = 9 / 3
x =     3
```

%NOTE UNLIKE C++ WE DO NOT WORRY ABOUT INT AND FLOAT VARIABLES RULES.

```
» x = 9.5 / 3
x =    3.1667

% with a variable;
» x = 15;
» y = x / 3
y =     5

» y = x / 4.5
y =    3.3333

% a more involved case
» a = 3;
» b = 5;
» c = 6;
» y = a /b + b / c
y =    1.4333
```

% HERE DIVISION IS DONE FIRST THEN ADDITION according to our priority rules

% WE CAN OVERRIDE THE ORDER by use of PARENTHESIS as in

```
» y=a/(b+b)/c
y =    0.0500
```

% The addition is done first followed dividing the sum into 'a;, followed with division of the result by 'c'; ie. **LEFT TO RIGHT operator evaluation.**

```
% More Priority rules illustrated
» a=2;
» b=4;
» c=6;
» d=7;
» y= a + b / c + d
y =    9.6667
```

% NOTE THE DIVISION WAS DONE FIRST- CONSIDER THE NEXT VARIATION

```
» y = a + b / (c + d)
y =    2.3077
```

% Parenthesis has the highest priority and was done first followed with the result of c+d divided into 'b' and finally the addition to 'a'.

```
% more parenthesis;
» y=( a + b) / ( c + d)
```

```
y =    0.4615
```

% in the above both sets of parenthesis are evaluated followed with the division of the sum of A+B by the sum of C+D.
% **WE MUST BE MINDFUL OF THE ORDER CALCULATIONS ARE CARRIED OUT AND BE CLEAR WE KNOW WHAT ALGEBRAIC EXPRESSION WE WANT TO EVALUATE. IF IN DOUBT about the order we can use PARENTHESES EVEN IF they are REDUNDANT so long as they clarify the mathematical operations needed.**

% **POWER OR EXPONENTIATION OPERATOR ^**

The ability to raise numbers or variables to a power is illustrated as follows.

```
» 2 ^ 2
ans =    4

» 3 ^ 2
ans =    9

» x = 7;
» y = x ^ 2
y =   49
```

NOTE**: IN MATLAB the exponentiation function ^ works only for variables that have one value.** We will see how to handle multi-valued variables which we call arrays or vectors further down.

```
» u = x ^ 2.5
u =  129.6418

» z=sqrt(x^2)
z =    7

» a=4;
» x=3;
» y=x^a
y =   81
```

ELEMENTARY PROGRAMMING (FUNCTIONS)

% INTRINSIC FUNCTIONS

Most functions found on a scientific calculator are in the MATLAB environment. Functions are used with parentheses containing the variable or value the function is to evaluate and the answer is usually returned and placed into a variable. Some functions illustrated below.

% There are **numerous functions in MATLAB** to see a list of them just click the **f_x prompt to the left of the command prompt >>**

% y=pi;   % the value of pi is a special constant and kept in variable 'pi'

% y = sin (x); Typical trigonometric functions are available but **x must be in radians**. Others are cos, tan, acos, asin, atan, sinh, asinh, etc.
Explore various values of these functions to get a sense how they work.
cos(0), cos(90) cos(pi) cos(pi/2) tan(pi/4)

%Try atan(1)   then hare multiply ans by 180/pi to see the **response was in radians. Remember to convert degrees to radians or vice versa we know   pi radians = 180 degrees**
% y = exp(x);   Exponential function. Also log (**natural log**), log10, abs, and sqrt are available.

% Many functions are built into the language others are programs residing on the "hard disk" of the computer. The programs reside on the disk as an entity called a file. The file contains the commands to execute the given function. Program files are known as "M -files" in MATLAB. We will return to files and programs later.

```
% Evaluations illustrated
» t = 25;
» z = sqrt(t)
z =     5
```

%The value in the parenthesis is called the **argument.**

```
% we illustrate a constant argument in the function with
» y = sqrt(99)
y =    9.9499

% additional functional illustrations follow
% use of a variable as argument
» x = 100;
» y = log(x)    % natural logarithm
y =    4.6052

» u = log10(x)
u =     2

» w = pi              % pi is a constant value
w =    3.1416

» z = sin(x)      % x is still a 100
z =   -0.5064

» x = 90;
» z = sin(x)
z =    0.8940

» x = 60;                % x stays 60 for a while
» y = sin(2 *x)
y =    0.5806
» y = 10 * sin ( 2 * x)
y =    5.8061

» y=10*sin(2*x+5)
y =   -6.1604

» y=10*sin(2*x+.5)
y =    8.9987

» y=10*sin(2*x+.5) + 5.* cos(3*x -2)
y =    6.6015
```

% pi is useful in trigonometric functions as in
% y=A*sin(2*pi*r)   etc.
% another useful function is raising values to a power which we saw for simple case we use the ^ symbol.
% we call upon the”power(a,b) function which raises a^b for a simple variable.

```
>>a=9
A=9
*>> z=power(a,2)
z= 81
```

% we will see an expand use below when we look at arrays or vectors.
% **If you need help about a particular function you can** type **'help item'** as in

```
» help log10
  log10(X) is the base 10 logarithm of the elements of X.
   Complex results are produced if X is not positive.
```

```
>>doc   will lead to numerous documentation on MATLAB
```

%note the fx link on the left of the prompt >>. It called a function browser. You might try the “fx' to get a list of various functions available.

% **One easily notices that many computer languages use similar constructs.**

SCIENTIFIC NOTATION OR EXPONENTIAL NOTATION (powers of ten representation)

% Very large and very small values are entered into the MATLAB environment similar to a calculator by using exponential notation. This notation uses the letter 'e' to indicate a power of 10, as illustrated below small powers are translated directly to the number on echo, others are kept in exponential notation (actually scientific notation with always one digit to the left of the decimal sign)

```
% EXPONENTIAL constants
» y=1.0e1
y =    10

» y=1e1
y =    10

» y=1e2
y =   100

» y=1e4
y =       10000

» y=1e10
y =  1.0000e+010

» y=1e-55
y =  1.0000e-055
```

% WE NOTE THAT THE 'e' MUST FOLLOW DIRECTLY THE NUMBER (**NO SPACES**) AS IN

```
» y=1.0 e1
y=1.0 e1
Error: Unexpected MATLAB expression.
```

% This last line is an error message, but MATLAB cannot see that only an extra space was inserted-the correct expression is y=1.0e1
Some error messages will make sense others only inform you that you must try to find out what you did wrong.
% The displayed type of number is called the **format.**
% Calculations are done in the memory to a large precision, but how the number is displayed can be changed or effected by its value as illustrated here.
% The normal mode is called format short and is gotten by default.

```
» x= 4/3
x =    1.3333
```

```
» y =2.456e-8
y =  2.4560e-008

» format short     % this 'format' command can change how numbers are shown
» x                % but by default we had it anyway
x =    1.3333
» y
y =  2.4560e-008

% to get all exponential representations of the numbers we use
» format short e
» x
x =  1.3333e+000

» y
y =  2.4560e-008

% for maximum decimal representation we use
» format long
» x
x =   1.33333333333333
       % note here 15 decimal places which is most likely system dependent.
» y
y =    2.456000000000000e-008

» format long e
» x
x =    1.333333333333333e+000
» y
y =    2.456000000000000e-008
% Try also  format short e to see what happens to x,y .
>> format short e
% We note that numbers with no decimal values are displayed as integers as in
% r = 1
r =     1
% and
» r=1.0
r =     1
% and
» r=1.000
r =     1
```

%   **ARROW KEYS AND MATLAB STATEMENTS**

% As you work in MATLAB each line you enter is remembered and can be recalled by the use of the up and down arrow keys. This permits you to execute a line again by going to it with your arrow keys. You are asked to try this in the tasks that follow

HOMEWORK LABORATORY TASKS (YOU RUN THESE ON THE COMPUTER and hand in!)

I suggest after each task looks successful you highlight what you want to ultimately print and copy to word, condense any long outputs, and when all tasks are done you can print the word document containing the following tasks. You can also print directly from the command window (highlight and print) but we don't want to waste too much paper.

NOTES: variables and constants are kept in memory in an area called the "WORKSPACE". The latter should be displayed in your initial setup. If not I recommend that you choose the "Default" option in the "Layout" icon.
Use >> clear to clear out data in memory and use "clc" or the right mouse click "clear command window" to start fresh. And also to get the print command which should be used sparingly. Using semi colon will prevent echo of the action that is not necessary to display which can minimize output to copy.

1. Initialize a variable to 4 and count by 3 ten times. What is the final value?

2. set up a conversion between degrees and radians. if x is in degrees than y will be radians. By using up arrow key to recall previously entered commands get the radian equivalent for 0,30,45,60,90,135,180,270 and 360 degrees.(ie. just keep changing x and recall the expression for y with the up arrow keys). Use the 'pi' function and your calculator to check results.

3. Using the same radian equivalents as in task 2 obtain the sine of each angle. Since commands are remembered you can bring them back with the up and down arrow keys. Check the results with your calculator.

4.(2 pts) Solve the position and velocity of a constantly accelerating motor cycle clocked at
a = 6.4 m/sec /sec, knowing that the time, t, from the starting position is 34.567 seconds.
Solve the position x and velocity v which are given by the formulas

$$x = 0.5\ at^2 \qquad v = at$$

5. Given a circle of radius 1.4 meters set up and solve for the diameter, the circumference and the area of the circle. Display the calculated numbers with maximum decimal values and use the 'pi' function.

6. Given a right triangle with sides equal to 1.234 and 7.89 find the hypotenuse of the triangle, as well as, its area. Use the sqrt function with the Pythagorean theorem to get the hypotenuse.

7. (3 pts.) Solve for the number of radioactive nuclei of Carbon-14 left, n, after a time of t =3000 years, with the starting number being ,n_0 = 1.5e6 and the formula $n = n_0\ e^{-(Lt)}$ with L = log(2)/5000 (the natural log here).
NOTE: 'e' in formula is the exponential function (exp() in MATLAB) not the exponent 'e' (powers of ten notation).

% Explore MATLAB by trying the following commands
demo
then explore the various choices for a deeper understanding
Useful help command
help
specific exploration
help plot
help exp